



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/620,534	07/16/2003	Hoi Chang	P00620-US-00 (19232.0003)	8981
22446	7590	06/27/2006	EXAMINER	
ICE MILLER LLP ONE AMERICAN SQUARE, SUITE 3100 INDIANAPOLIS, IN 46282-0200			KIM, JUNG W	
			ART UNIT	PAPER NUMBER
			2132	

DATE MAILED: 06/27/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/620,534	CHANG ET AL.	
	Examiner	Art Unit	
	Jung Kim	2132	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 05 May 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 67-72, 78-88, 90-92, 94-96 and 99-118 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 67-72, 78-88, 90-92, 96 and 99-118 is/are rejected.
- 7) ☒ Claim(s) 94 and 95 is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This Office action is in response to the amendment filed on May 5, 2006.
2. Claims 67-72, 78-88, 90-92, 94-96 and 99-118 are pending.
3. Claims 1-66, 73-77, 89, 93, 97 and 98 are canceled.
4. Claims 101-118 are new.

Continued Examination Under 37 CFR 1.114

5. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on May 5, 2006 has been entered.

Response to Arguments

6. Applicants' arguments have been fully considered but they are not persuasive. Regarding applicant's argument that claims 67-72 and 78-87 fulfill the written description requirement, examiner respectfully disagrees. The examiner's basis for the 112/1st paragraph is outlined below under the section "Claim Rejections 35 USC 112." Examiner does not find Applicants arguments to be persuasive because they do not specify any particular portion of the original Specification that identifies the particulars of

the claimed invention under rejection. In fact, applicant's assertion of compliance with the written description requirement is only based on an unsupported allegation of compliance. In particular, applicant cites:

In Applicant's response mailed March 7, 2006 ("Response"), responding to the Final Action, Applicant's point out portions of the specification that would show to one of skill in the art that, as of the filing date, Applicants were in possession of the invention as claimed.

In the Advisory Action, the Examiner states that "there is a general mention of selecting and adding a silent guard variable to the software program in the Specification ... However, none of the other portions of the Specification indicated by the Applicant illustrate the remaining steps ... All the portions Applicant points to ... illustrates examples of a silent guard in use. These are two distinct inventions." (Advisory Action, page 2, lines 3-9)

Applicants respectfully submit that Applicant's specification would show to one of ordinary skill in the art that, as of the filing date, Applicants were in possession of the invention as claimed. ,The Examiner finds that portions of the Specification pointed out in the Response "illustrates examples of a silent guard in use" (Advisor Action, page 2, lines 8-9). Applicants submit that, from examples of a silent guard in use and the accompanying explanation provided in the Specification, one of ordinary skill in the art would know Applicant's were in possession of the invention as claimed at the time of filing. (Remarks, pg. 18, paragraphs 1-3)

This argument does not provide any facts to support their contention. Merely stating that "Applicants submit that, from examples of a silent guard in use and the accompanying explanation provided in the Specification, one of skill in the art would know Applicants were in possession of the invention as claimed at the time of filing" does not adequately identify why this is the case, nor does it provide evidence or

Art Unit: 2132

rational why the rejection is not valid. Hence, claims 67-72 and 78-87 remain rejected under 112/1st paragraph.

Regarding applicant's argument that Collberg teaches away from the claimed inventions, examiner disagrees. In particular, applicant's basis for their argument is that "Collberg requires that the target program terminates and produces the same output as the source program, i.e., that the target program executes properly," pg. 21, 1st full sentence. However, this is a requirement of Collberg for devising a proper transformation from non-obfuscated code to obfuscated code, not a requirement of the software after the obfuscated code is being used by a user as implied by the applicants. In applicant's claim 67, the limitation in question, "revising the selected computation to be dependent on the runtime value of the silent guard variable, such that the software program executes improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard" only defines making the code dependent on the runtime value of the inserted code such that a consequence is incurred based on an intended use after the code is tamperproofed (i.e. the software program executes improperly when the runtime value of the silent guard variable is modified (directly or indirectly) from the expected value by a third party completely independent from the claimed invention) In the case of any of Collberg's examples, any of the obfuscated code would execute improperly if the runtime value of any of the variables of the obfuscated code is modified from the expected value. Hence, the teachings of Collberg do not teach away from applicant's claimed inventions.

Applicant's argument that Collberg does not teach computing the runtime value of the silent guard variable using a mathematical expression using both the runtime value and the expected value of a program variable as recited (Remarks, pg. 22) is not persuasive. Collberg's example illustrates a transformation wherein selected program variables are chosen (A, B, C) and this runtime value of the these selected program variables are embodied as variables a1, a2, b1, b2, c1 and c2. Moreover, the silent guard "x" is assigned the output of the various functions as expressed in steps 5', 7' and 8', wherein the expected values of these selected program variables are included as a precondition of the functions and operators (i.e. the branch conditions of steps 8', 9' and 10' are dependent on the expected values of A, B and C).

The remaining arguments against the prior art rejections of the amended claims are moot in view of the new rejections under Johnson alone and also in view of Collberg.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

8. Claims 67-72 and 78-87 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject

matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

9. As per claims 67-72 and 78-87, the Specification discloses a general method for adding tamper resistance to software program (fig. 26 and related text); however, it does not describe an explicit method for adding tamper resistance to a software program using silent guards as recited in these claims. Claims 67-72 and 78-87 define an explicit method that includes steps, "adding a silent guard ...", "selecting a computation in the software program ...", "determining an expected value of the silent guard variable at the execution point of the selected computation ...", "setting the runtime value of the silent guard variable ..." and "revising the selected computation to be dependent on the runtime value of the silent guard ..." (see claim 67); "selecting a program variable ...", "selecting a computation in the software program ...", "determining an expected value of the program variable at the point of execution of the selected computation ...", and "revising the selected computation to be dependent on the runtime value of the program variable ..." (see claim 78); "selecting a program block containing a step necessary for proper execution of the software program the program block comprising at least one program instruction ...", "selecting a silent guard for the program block ...", "determining the expected value of the silent guard at the start of execution of the program block ...", and "installing a branch instruction dependent on the silent guard in the software program ..." (see claim 82). However, none of these

methods to add a silent guard are disclosed in the Specification. As stated above, the specification only discloses a general method of adding tamper resistance to a software program (fig. 26 and related text), and the portions of the specification describing silent guards only refer to silent guards that have already been added to the software programs (pg. 28, line 4-pg. 34, line 10 and pg. 84, line 22-pg. 88, line 6).

10. In addition, the limitation of claim 70 and 79, "selecting a constant value used in the selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime value of the [silent guard/program] variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable" are not described in the specification.

Claim Rejections - 35 USC § 102

11. Claims 78, 80-87, 91, 92, 96, 99-118 are rejected under 35 USC 102(b) as being anticipated by Johnson USPN 5,748,741 (hereinafter Johnson).

12. As per claim 78, Johnson discloses a computer implemented method for adding tamper resistance to a software program, the method comprising;

- a. selecting a program variable in the software program (col. 10:29);
- b. selecting a computation in the software program (col. 10:16; 12:11-14: the code following insertion of the check);

- c. determining an expected value of the program variable at the point of execution of the selected computation (col. 12:10); and
- d. revising the selected computation to be dependent on the runtime value of the program variable, such that the software program executes incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable (col. 12:11-14).

13. As per claim 80, Johnson further discloses the step of revising the selected computation comprises: selecting a computation variable used in the selected computation; determining an expected value of the computation variable at the point of execution of the selected computation; and replacing the computation variable with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the program variable is equal to the expected value of the program variable (col. 10:42-11:18).

14. As per claim 81, Johnson further discloses the step of revising the selected computation comprises: inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation (col. 10:42-58).

15. As per claim 82, Johnson discloses a computer implemented method for adding tamper resistance to a software program, the method comprising:

- e. selecting a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction (col. 10:16; 12:11-14: the code following insertion of the trap);
- f. selecting a silent guard for the program block; determining the expected value of the silent guard at the start of execution of the program block (col. 12:12); and
- g. installing a branch instruction dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch instruction causes the program block to be skipped, causing the software program to execute improperly (col. 12:13).

16. As per claim 83, Johnson further discloses the step of selecting a silent guard comprises: adding a silent guard variable to the software program; using the silent guard variable as the silent guard; and installing an initialization instruction for the silent guard variable that executes prior to the branch instruction in the software program, the initialization instruction setting the silent guard variable equal to the expected value of the silent guard (col. 11:18).

Art Unit: 2132

17. As per claim 84, Johnson further discloses the step of selecting a silent guard comprises: selecting a program variable in the software program; and using the program value as the silent guard (col. 10:29 and 10:42-11:7).

18. As per claim 85, Johnson further discloses the step of selecting a silent guard comprises:

h. selecting an insertion point in the software program; selecting a program variable in the software program; determining the expected value of the program variable at the insertion point; making the runtime value of the silent guard dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point (col. 10:29-11:18).

19. As per claim 86, Johnson further discloses the step of making the runtime value of the silent guard dependent on the runtime value of the program variable, comprises:

i. installing a mathematical computation that includes the runtime value of the program variable, such that the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point; and setting the silent guard equal to the result of the mathematical computation (col. 10:54-11:4; 11:17).

20. As per claim 87, Johnson further discloses the silent guard computation uses both the expected value of the program variable and the runtime value of the program variable (col. 10:42-58).

21. As per claim 96, Johnson discloses a recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

- j. a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction; (col. 10:16; 12:11-14: the code following insertion of the check)
- k. a silent guard having an expected value at the start of the execution of the program block (12:11);
- l. a program variable having an expected value at an insertion point in the software program; the runtime value of the silent guard being dependent on the runtime value of the program variable at the insertion point, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point; (10:29-11:18) and
- m. a branch instruction dependent on the silent guard, the branch instruction being separated from the insertion point by a plurality of program instructions of the software program (12:10-12, lines 22-23 and 25-27);

n. wherein the branch instruction will cause the program block to be skipped if the runtime value of the silent guard is not equal to the expected value of the silent guard with will cause the software program to execute improperly. (12:14)

22. As per claim 99, Johnson further discloses the runtime value of the silent guard is made dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, wherein the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point. (10:29-11:18)

23. As per claim 100, Johnson further discloses the runtime value of the silent guard is also dependent on the expected value of the program variable at the insertion point. (col. 10:42-58).

24. As per claim 101, Johnson further discloses the program variable is used in more than one instruction of the software program, and the last instruction to use the program variable before the point of execution of the selected computation during execution of the software program is separated from the point of execution of the selected computation by a plurality of program instructions not using the program variable. (col. 12:22-24 and lines 25-27)

25. As per claim 102, Johnson further discloses the installation point of the initialization instruction for the silent guard variable is separated from the installation point of the branch instruction by a plurality of program instructions of the software program. (col. 12:22-24 and lines 25-27)

26. As per claim 103, Johnson further discloses the insertion point is separated from the installation point of the branch instruction by a plurality of program instructions of the software program. (col. 12:22-24 and lines 25-27)

27. As per claim 104, Johnson further discloses the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program. (col. 12:22-24 and lines 25-27)

28. As per claim 105, Johnson further discloses the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program. (col. 12:22-24 and lines 25-27)

29. As per claim 106, Johnson discloses a recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

- o. a silent guard variable having an expected value at a first dependency point in the software program (col. 10:40:49; silent guard variables are t, u, v and w);
- p. a program variable having an expected value at a second dependency point in the software program (col. 10:16; 11:18; "s");
- q. a mathematical computation that includes the run time value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point (col. 10:42-11:14; the runtime value of silent guards is t, u, v and w; the expected term is $2t + v + w$ and $2u + v + w$);
- r. wherein the runtime value of the program variable at the second dependency point is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable at the first dependency point, such that the runtime value of the program variable at the second dependency point will not equal the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable at the first dependency point does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly. (10:62; 11:4 and line 18; s is dependent on whether the runtime value of silent guard t, u, v and w equal the expected value of the silent guard variable x and y [if $y == 2t + v + w$; $x == 2u + v + w$; then $s == (x + y)/4$])

30. As per claim 107, Johnson further discloses the first dependency point is separated from the second dependency point by a plurality of program instructions. (col. 12:22-24 and lines 25-27)

31. As per claim 108, Johnson further discloses the tamper resistant software program further comprising an instruction setting the value of the program variable at the second dependency point equal to a function of the runtime value of the program variable and the result of the mathematical computation. (the runtime value of the program variable = $(x + y)/4$; furthermore, $y == 2t + v + w$; $x == 2u + v + w$;))

32. As per claim 109, Johnson discloses a recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

- s. a program block for causing improper execution of the software program, the program block including at least one program instruction (col. 12:14);
- t. a silent guard in the software program having an expected value at the start of execution of the program block (12:11); and
- u. a branch instruction in the software program dependent on the runtime value of the silent guard, wherein the branch instruction will cause the program block to be executed if the runtime value of the silent guard is not equal to the

expected value of the silent guard, causing the program to execute improperly (12:11-14).

33. As per claim 110, Johnson further discloses the tamper resistant software program further comprising a program variable having an expected value at an insertion point (col. 10:29); wherein the runtime value of the silent guard is dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point. (col. 10:42-11:18; 12:11)

34. As per claim 111, Johnson further discloses the insertion point is separated from the branch instruction by a plurality of program instructions. (col. 12:22-24 and lines 25-27)

35. As per claim 112, Johnson further discloses the runtime value of the silent guard is dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, wherein the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point. (col. 10:42-49 and line 62; 11:3 and line 18)

Art Unit: 2132

36. As per claim 113, Johnson further discloses the mathematical computation includes an expected term, the expected term being set based on the expected value of the program variable at the insertion point. (col. 10:42-11:14)

37. As per claims 114-118, they are method claims corresponding to claims 109-113, and they do not teach or define above the information claimed in claims 109-113. Therefore, claims 114-118 are rejected as being unpatentable as being anticipated by Johnson over for the same reasons set forth in the rejections of claims 109-113.

38. As per claims 91 and 92, they are method corresponding to claims 106-108, and they do not teach or define above the information claimed in claims 106-108. Therefore, claims 91 and 92 are rejected as being unpatentable as being anticipated by Johnson over for the same reasons set forth in the rejections of claims 106-108.

Claim Rejections - 35 USC § 103

39. Claims 67-72, 79, 88 and 90 rejected under 35 U.S.C. 102(a) as being unpatentable over Collberg "A Taxonomy of Obfuscation Transformations" (hereinafter Collberg) in view of Johnson.

40. As per claim 67, Collberg discloses a computer implemented method for adding tamper resistance to a software program (pg. 18, section 7.1.3; pg. 19, fig. 18(a-e)), the method comprising:

- v. adding a silent guard variable to the software program (pg. 19, fig. 18(a-c), fig. 18(e), the variable "x");
- w. selecting a computation in the software program (pg. 19, fig. 18(e), steps 1-10);
- x. determining an expected value of the silent guard variable at the execution point of the selected computation (pg. 19, fig. 18(e), steps 5', 7' and 8': assignment of x;
- y. setting the runtime value of the silent guard variable to the expected value of the silent guard variable in the software program at a silent guard insertion point; and
- z. revising the selected computation to be dependent on the runtime value of the silent guard variable, such that the selected computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable (pg. 19, fig. 18(e), steps 5', 7' and 8': assignment of c1 and c2.

41. In the example of figure 18, Collberg does not expressly disclose the silent guard insertion point being separated from the execution point of the selected computation by a plurality of program instructions of the software program. However, in a different section, Collberg discloses inserting dead or irrelevant code into a basic block, and also interleaving methods to introduce transformations to increase the complexity of the code and hence enhance the level of obfuscation. (pg. 11, section 6.2.1 "Insert Dead or Irrelevant Code"; pg. 15, section 6.3.2, "Interleave Methods") Therefore, it would be

obvious to one of ordinary skill in the art at the time the invention was made for the silent guard insertion point to be separated from the execution point of the selected computation by a plurality of program instructions of the software program. One would be motivated to do so to increase the level of obfuscation of the code as taught by Collberg, *ibid*.

42. Moreover, in the example of figure 18, Collberg discloses that the selected computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable (as noted above), but does not disclose that the software executes improperly. Johnson discloses a method of tamperproofing software by introducing cascades and intertwining blocks. One technique taught by Johnson relevant to the limitation of the instant claim is a feature of checking whether the computed values are different from the expected values, and if not then a TamperFlag is set and a timer is started; when the timer expires, operation of the software program terminates at a trap code to prevent tampering of the software code. (col. 12:13-15) Therefore, it would be obvious to one of ordinary skill at the time the invention was made for the software program to execute improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable to prevent tampering of the software. The aforementioned cover the limitations of claim 67.

43. As per claim 68, the rejection of claim 67 under 35 USC 103(a) as being unpatentable over Collberg in view of Johnson is incorporated herein. (*supra*) Collberg

Art Unit: 2132

further discloses: selecting a program variable in the software program; determining an expected value of the selected program variable at a dependency point in the software program; and making the runtime value of the silent guard variable dependent on the runtime value of the selected program variable at the dependency point (pg. 18, section 7.1.3, especially 6th paragraph; pg. 19, fig. 18(a-e), A, B and C are translated into a1, a2, b1, b2, c1, c2, and x is the runtime value of the silent guard).

44. As per claim 69, the rejection of claim 68 under 35 USC 103(a) as being unpatentable over Collberg in view of Johnson is incorporated herein. (supra) Collberg further discloses the step of making the value of the silent guard variable dependent on the runtime value of the selected program variable comprises: computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point (pg. 19, fig. 18(e), steps 5', 7' and 8').

45. As per claim 70, the rejection of claim 67 under 35 USC 103(a) as being unpatentable over Collberg in view of Johnson is incorporated herein. (supra) In addition, Collberg further discloses a method of converting static variables into procedural data as an obfuscation technique. In an example, a constant string within a selected computation is replaced by a function wherein a parameter of the function determines the string generated by the function (pgs. 18-19, sections 7.1.4 and fig. 19). Furthermore, this function is effectively a mathematical expression dependent on the

variable parameter and evaluates to the constant string value if the run time value of the variable parameter is equal to the expected value of the variable parameter. Therefore, it would be obvious to one of ordinary skill in the art at the time the invention was made for the step of revising the selected computation to comprise the steps: selecting a constant value used in selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime of the silent guard variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable. One would be motivated to incorporate these steps since it prevents unscrupulous users from identifying the operation of the software (Collberg, *ibid*). The aforementioned cover the limitations of claim 70.

46. As per claim 71, the rejection of claim 67 under 35 USC 103(a) as being unpatentable over Collberg in view of Johnson is incorporated herein. (*supra*) Collberg further discloses the step of revising the selected computation comprises: selecting a computation variable used in the selected computation; determining an expected value of the computation variable at the execution point of the selected computation; and replacing the computation variable with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable (pg. 19, fig. 18(e), steps 5', 7' and 8').

47. As per claim 72, the rejection of claim 67 under 35 USC 103(a) as being unpatentable over Collberg in view of Johnson is incorporated herein. (supra) Collberg further discloses the step of revising the selected computation comprises: inserting a mathematical expression including the runtime value of the silent guard variable and the expected value of the silent guard variable into the selected computation (pg. 19, fig. 18(e), steps 5'-10')

48. As per claim 79, the rejection of claim 78 under 35 USC 102(b) as being anticipated by Johnson is incorporated herein. (supra) Johnson does not disclose for the step of revising the selected computation to comprise the steps: selecting a constant value used in selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime of the program variable, such that the mathematical expression evaluates to the constant value if the runtime value of the program variable is equal to the expected value of the program variable. Collberg discloses a method of converting static variables into procedural data as an obfuscation technique. In an example, a constant string within a selected computation is replaced by a function wherein a parameter of the function determines the string generated by the function (pgs. 18-19, sections 7.1.4 and fig. 19). Furthermore, this function is effectively a mathematical expression dependent on the variable parameter and evaluates to the constant string value if the run time value of the variable parameter is equal to the expected value of the variable parameter. Therefore, it would be obvious to

one of ordinary skill in the art at the time the invention was made for the step of revising the selected computation to comprise the steps: selecting a constant value used in selecting computation; and replacing the constant value with a mathematical expression that is dependent on the runtime of the program variable, such that the mathematical expression evaluates to the constant value if the runtime value of the program variable is equal to the expected value of the program variable. One would be motivated to incorporate these steps since it prevents unscrupulous users from identifying the operation of the software (Collberg, *ibid*). The aforementioned cover the limitations of claim 79.

49. As per claims 88 and 90, the rejection of claims 67-72 as being unpatentable over Collberg in view of Johnson is incorporated herein. (*supra*) In addition, Collberg discloses inserting dead or irrelevant code into a basic block, and also interleaving methods to introduce transformations to increase the complexity of the code and hence enhance the level of obfuscation. (pg. 11, section 6.2.1 "Insert Dead or Irrelevant Code"; pg. 15, section 6.3.2, "Interleave Methods") Therefore, it would be obvious to one of ordinary skill in the art at the time the invention was made for the dependency point to be separated from the execution point by a plurality of program instructions. One would be motivated to do so to increase the level of obfuscation of the code as taught by Collberg, *ibid*. The aforementioned cover the limitations of claims 88 and 90.

Allowable Subject Matter

Claims 94 and 95 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Jakubowski et al. USPN 7,054,443.

Homing et al. US Publication No. 20050210275.

Communications Inquiry

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jung W. Kim whose telephone number is 571-272-3804. The examiner can normally be reached on M-F 9:00-5:00.

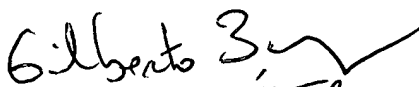
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



June 22, 2006.

Jung W Kim
Examiner
Art Unit 2132



GILBERTO BARRON JR.
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100